Doc is in updating state

GIT & GIT hub quick notes

**What is Git?**

Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

**It is used for:**

* Tracking code changes
* Tracking who made changes
* Coding collaboration

**What does Git do?**

* Manage projects with Repositories
* Clone a project to work on a local copy
* Control and track changes with Staging and Committing
* Branch and Merge to allow for work on different parts and versions of a project
* Pull the latest version of the project to a local copy
* Push local updates to the main project

Working with Git

* Initialize Git on a folder, making it a Repository **(> git init** )
* Git now creates a hidden folder to keep track of changes in that folder
* When a file is changed, added or deleted, it is considered modified
* You select the modified files you want to Stage (> git add .)
* The Staged files are Committed, which prompts Git to store a permanent snapshot of the files
* Git allows you to see the full history of every commit.
* You can revert back to any previous commit.
* Git does not store a separate copy of every file in every commit, but keeps track of changes made in each commit!

Why Git?

* Over 70% of developers use Git!
* Developers can work together from anywhere in the world.
* Developers can see the full history of the project.
* Developers can revert to earlier versions of a project.

What is GitHub?

* Git is not the same as GitHub.
* GitHub makes tools that use Git.
* GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.

Git Branch

Branches allow you to work on different parts of a project without impacting the main branch.

When the work is complete, a branch can be merged with the main project.

edit the code directly without impacting the main branch

**create new branch**

git branch branch2

**Note:** Using the -b option on checkout will create a new branch, and move to it, if it does not exist

**to check the branch**

git branch

branch2

\* master

\* beside master specifies that we are currently on that branch.

To switch between the branchs

git checkout branch2

GIT basic commands

|  |  |
| --- | --- |
| Version | git --version |
| Configure Git | git config --global user.name "saied"  git config --global user.email "xyz@mail" |
| Initialize git | git init |
| Check git status | git status, git status --short |
| Adding file to staging env | git add filename or git add . |
| move file from stage to commit | git commit -m “commit msg ” |
| Git Commit without Stage | git commit -a -m “commit msg ” |
| Git commit log | git log, git log –oneline, --all |
| Create new branch | git branch branch-namegit checkout -b branch-name (create and switch)git switch -c <new-branch-name> |
| Switch branches | git checkout branch-name |
| Check branches | git branch |
| Delete branch | git branch -D branch-name |
| Revert modifications | git stash |
| Merge branch | git merge branch-name |
| Rename branch | $ git branch --move old-branch-name new-branch-name |
|  | Git rebase |
| Undo commit | Git revert (Commit id) revert changes made by that commit |
| Git reset  Git reset –hard id | Will delete all commits after the mentioned commit |
| .gitignore file | Contains file and folder that should not be staged or commited |

**git stash**

Use git stash when you want to record the current state of the working directory and the index, but want to go back to a clean working directory. The command saves your local modifications away and reverts the working directory to match the HEAD commit.

The modifications stashed away by this command can be listed with git stash list, inspected with git stash show, and restored (potentially on top of a different commit) with git stash apply. Calling git stash without any arguments is equivalent to git stash push. A stash is by default listed as "WIP on **branchname** …​", but you can give a more descriptive message on the command line when you create one.

**Establishing connection with remote repo:**

$ Git add origin https// repo url

Create the push connection

$ git push --set-upstream origin master

after using this once if we make any changes in local repo we simply has to use

$ git push

Method 2

# git remote set-url origin <https://saiedbilal@github.com-----url> continue

This command will instruct git that saiedbilal is interacting with github

#git push origin main

We will get prompt to provide password to authenticate.

We need to use personal access token

Setting>dev setting>personal access token >generate

Check repo check box

Git hub readme is written in markdown .md

GITHUB

|  |  |
| --- | --- |
| adding a remote repository | git remote add origin *URl*   * Origin is local identifier |
| Verify new remote | git remote -v |
|  | git config -l |
| Cloning repo | Git clone repo url |
| update local from remote | Git pull / git pull url |
| update remote from local | Git push |
| Renaming a remote repository | git remote rename origin newname |
|  |  |
| Push new branch from local to remote | Git push origin new\_branch |
| New branch from remote to local | Git pull origin branch\_name |
| Update local branch from remote | Git pull origin |
| Update hub from remote | Git push origin |
|  |  |
|  |  |
|  |  |